

***DONE BY:***

***SAHITHYA BALAKRISHNAN AND***

***S.AISHWARYA***

***Financial Accounts Management System***

COMPUTER SCIENCE PROJECT

****

**HIRANANDANI UPSCALE SCHOOL**

**BONA FIDE CERTIFICATE**

**This is to certify that --------------------------------------------------, student of class XII has successfully completed the project entitled --------------------------------------------------------------- during the year 2017 – 2018 in partial fulfilment of computer science practical examination.**

**Teacher-in-Charge: ------------------------**

**Register No: ------------------------**

**Date: ------------------------**

**Principal Internal Examiner External Examiner**

Table of contents:

|  |  |  |
| --- | --- | --- |
| **SR. NO.** | **TITLE** | **PAGE NO.** |
| 1. | Acknowledgement | 1 |
| 2. | Introduction | 2 |
| 3. | System Requirements | 2 |
| 4. | Flow Chart | 3 |
| 5 | Header Files Used | 4 |
| 6. | Modules of the Application | 4 |
| 7. | Modules Description | 4 |
| 8. | Source Code | 7 |
| 9. | Sample Outputs | 59 |
| 10. | Future Enhancements | 64 |
| 11. | Conclusion | 64 |
| 12. | References | 65 |

ACKNOWLEDGEMENT

I take this opportunity to express my gratitude and respect to **Mrs. Usha Ramakrishnan**, the vice Principal of Hiranandani Upscale School Egattur Chennai Tamil Nadu. Her dedication and keen interest and above all the overwhelming gratitude to help the students through timely advices and meticulous scrutiny have helped me to a very great extent to accomplish this task.

It is a genuine pleasure to express my deep sense of gratitude to **Mrs. Selvapriya Vadhana**, Head of Department of Computer Science Hiranandani Upscale School Egattur Chennai Tamil Nadu, for her keen interest on me at every stage of my project. Her prompt inspirations, scientific approach, timely suggestions with kindness, enthusiasm and dynamism have enabled me to complete my project.

Finally I would like to place my utmost regard to my family and classmates for aiding me throughout my venture.

INTRODUCTION

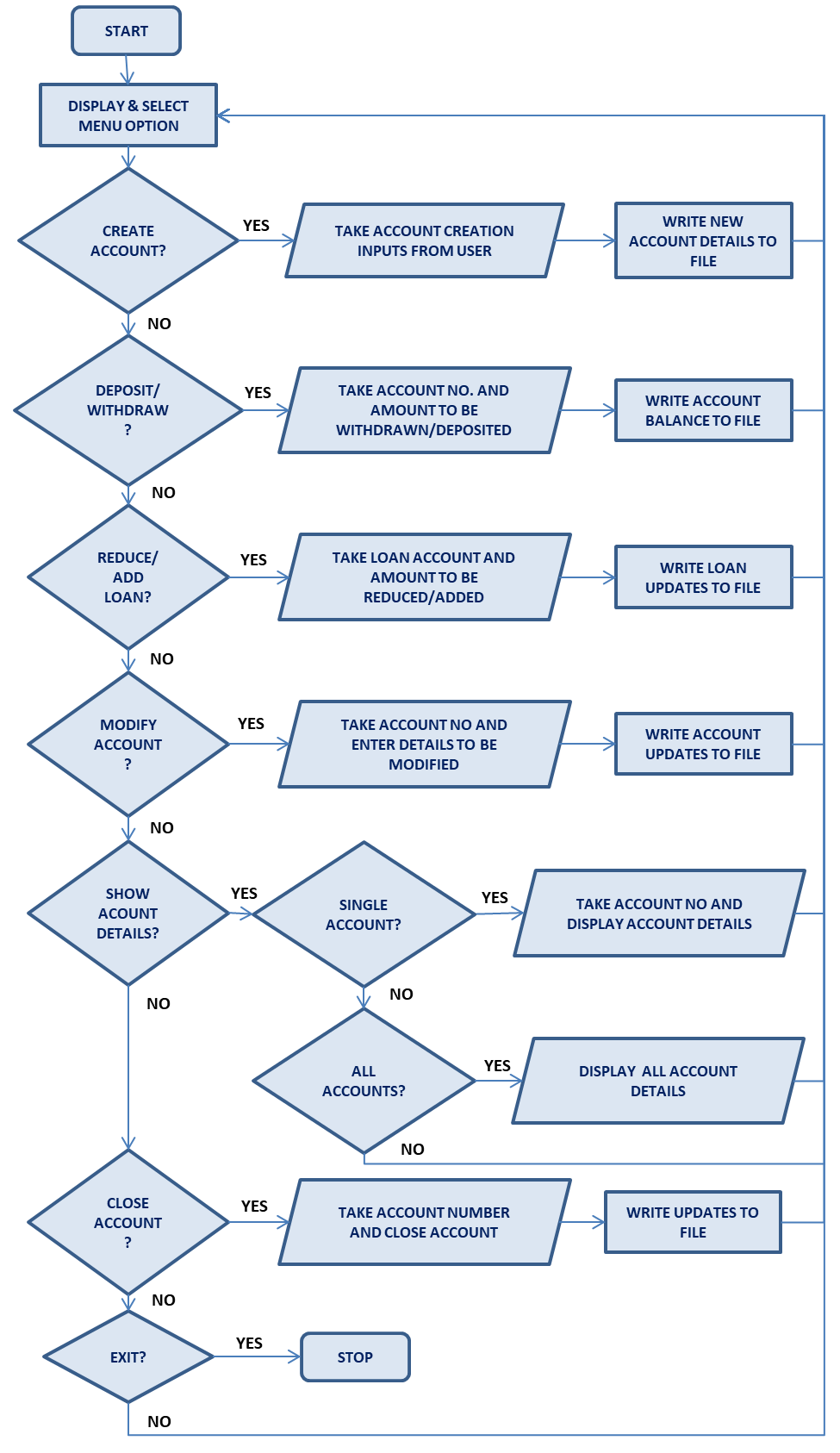
The banking industry works by collecting money from customers and lending it to borrowers. It is very important for the banks to maintain the accounts and transactions made in the accounts accurately. It will be very difficult to do this manually. A computerised application will help the banks to easily maintain their accounts and the transaction details.

The **Financial Accounts Management System** is a program which has been developed for banks to easily create and maintain their customers’ financial accounts like savings accounts, current accounts and loan accounts. This application will help the bank users to create new accounts, withdraw and deposit amount from existing accounts, close accounts etc. The application will also help the bank user to run reports to see the details of existing accounts.

The application has been developed C++, which is an object orient programming language. We have leveraged the features of C++ such as classes, inheritance, polymorphism, data abstraction and encapsulation in the program. This will help in the easy maintenance and enhancement of the application.

System requirements:

* Front end: Turbo C++
* Operating System: Windows 7

FLOW CHART:

HEADER FILES USED:

*#include<iostream.h>  
#include<conio.h>  
#include<string.h>  
#include<fstream.h>  
#include<ctype.h>   
#include<iomanip.h>*

MODULES OF THE APPLICATION:

The program consists of the following modules:

1. Creating an account
2. Depositing/Adding amount
3. Removing/Adding loan
4. Modify account details
5. Show account details
6. All account holder list
7. Closing an account

MODULES DESCRIPTION:

1. **Creating an account:**

The functionality of the module is to enable the user to create an account (transaction/loan) by entering details like name, phone number, initial deposit etc. Once the account has been created, the program assigns an account number to the user and the account details are written to the file where all the account and transaction details are stored. The account will be either transaction or loan type.

1. **Depositing/Withdrawing amount:**

The functionality of the module is to enable the user to deposit/ withdraw amount from their account (transaction account). The user enters the account number and then enters the amount to be deposited/ withdrawn. In case of withdrawing amount, if the balance amount after withdrawal is below the minimum balance amount, then the program displays a warning and the transaction does not take place. Otherwise, the account balance is updated and written to the file.

1. **Removing/Adding loan:**

The functionality of the module is to enable users to remove/ add loan amount to their account. The user enters the account number and then enters the amount to be removed/ added to the loan. The updated loan amount is then written to the account number in the file.

1. **Modify account details:**

The functionality of the module is to enable users to modify the account details (account holder name, nominee/guarantor and contact number). The user enters the type of account (transaction/loan) and the account number. The user then makes the required changes to their account details and the updates are saved to the file.

1. **Show account details:**

The functionality of the module is to display the account details of the account number entered by the user. The user first enters the account type (transaction/loan) and then enters the account number. The account details of the entered account number are displayed.

1. **All account holder list:**

The functionality of the module is to display all the accounts present in the program. The user enters the account type (transaction/loan/both). Depending on the option selected, all the accounts of that type are displayed.

1. **Closing an account:**

The functionality of the module is to enable the user to close an account. The user first enters the account type (transaction/loan) and then the account number. The account is then closed and the status of the account becomes ‘Inactive/Closed’. The updates are saved to the file. No transactions can take place in a closed account.

SOURCE CODE:

/\*----------------------------------------------------------------------

HEADER FILES USED

----------------------------------------------------------------------\*/

#include<iostream.h>

#include<conio.h>

#include<string.h>

#include<fstream.h>

#include<ctype.h>

#include<string.h>

#include<iomanip.h>

/\*----------------------------------------------------------------------

Declaration of functions to be used in the account and derived classes

----------------------------------------------------------------------\*/

int get\_next\_txnacc(); // Function to return the next account number to be used for a new transaction account

int get\_next\_loanacc(); // Function to return the next account number to be used for a new loan account

/\*----------------------------------------------------------------------

Definition of class: account

Type of Class: Base Class

Description: This class hold the basic account details like:

Account number, name, nominee, type, contact number, active status.

----------------------------------------------------------------------\*/

class account

{

public:

char acc\_name[30];

int acc\_no;

int is\_active;

char contact\_no[11];

};

/\*----------------------------------------------------------------------

Definition of class: txn\_account

Type of Class: Derived Class; privately inherited

Inherited from the calss: account

Description: This class is privately inherited from account base class.

Holds the saving or current account details and related functions

----------------------------------------------------------------------\*/

class txn\_account:private account

{

private:

int deposit;

char acc\_type;

char nominee[30];

public:

void create\_account();

void modify\_account();

void deposit\_amount(int);

int withdraw\_amount(int);

void show\_acc\_details();

int get\_acc\_number();

void get\_acc\_name(char \*);

char get\_type ();

int acc\_active ();

int get\_balance();

void disable\_account();

};

/\*----------------------------------------------------------------------

Definition of class: loan\_account

Type of Class: Derived Class; privately inherited

Inherited from the class: account

Description: This class is privately inherited from account base class.

Holds the loan account details and related functions

----------------------------------------------------------------------\*/

class loan\_account:private account

{

private:

int loan\_balance;

char guarantor[30];

public:

void create\_account();

void modify\_account();

void reduce\_loan\_balance (int);

void increase\_loan\_balance (int);

void show\_acc\_details();

int get\_acc\_number();

void get\_acc\_name(char \*);

char get\_type ();

int acc\_active ();

int get\_balance();

void disable\_account();

};

/\*----------------------------------------------------------------------

Definition of functions of the class: txn\_account

----------------------------------------------------------------------\*/

//FUNCTION TO POPULATE THE TRANSACTION ACCOUNT DETAILS

void txn\_account::create\_account()

{

//Assign account number automatically

acc\_no = get\_next\_txnacc();

//The account is set to active by default

is\_active = 1;

// Get the name and nominee details

cout<<"\nENTER THE ACCOUNT HOLDER NAME: ";

cin.ignore();

cin.getline(acc\_name,30);

cout<<"\nENTER THE NOMINEE FOR THE ACCOUNT: ";

cin.clear();

cin.getline(nominee,30);

//Get the contact numner and validate that it is all digits

int contact\_ok = 0;

char tmp\_contact[10];

while (contact\_ok== 0)

{

cout<<"\nENTER CONTACT PHONE NUMBER (10 digits): ";

cin.clear();

cin >> tmp\_contact;

contact\_ok = 1;

for (int i=0; i < 10; i++)

{

if(!(isdigit(tmp\_contact[i] )))

{

cout << "PLEASE ENTER A VALID PHONE NUMBER - ONLY DIGITS 0 to 9!" << endl;

contact\_ok = 0;

break;

}

}

}

strcpy(contact\_no, tmp\_contact);

contact\_no[10] = '\0';

//Get the account type - Saving or Current

acc\_type = 'x';

while (acc\_type != 'S' && acc\_type != 'C')

{

cout<<"\nENTER THE ACCOUNT TYPE(S=SAVING; C=CURRENT):";

cin >> acc\_type;

acc\_type = toupper(acc\_type);

}

//Get the initial deposit amount based on account type

if (acc\_type == 'S')

{

deposit = 0;

while ( deposit < 500)

{

cout<<"ENTER THE INITIAL DEPOSIT AMOUNT(Min Rs. 500): ";

cin >> deposit;

}

}

else

{

deposit = 0;

while ( deposit < 1000)

{

cout<<"ENTER THE INITIAL DEPOSIT AMOUNT(Min Rs. 1000): ";

cin >> deposit;

}

}

}

//FUNCTION TO MODIFY TRANSACTION ACCOUNT DETAILS

void txn\_account::modify\_account()

{

char option;

while(option !='1' && option !='2' && option !='3' )

{

clrscr();

cout<<"\nOPTIONS TO MODIFY THE ACCOUNT:";

cout<<"\n1.CHANGE ACCOUNT HOLDER NAME";

cout<<"\n2.CHANGE NOMINEE NAME";

cout<<"\n3.CHANGE CONTACT NUMBER";

cout<<"\nENTER OPTION:";

cin>>option;

}

switch(option)

{

case '1':

cout<<"CURRENT NAME OF THE ACCOUNT HOLDER: " << acc\_name;

cout<<"\nENTER THE UPDATED NAME: ";

cin.ignore();

cin.getline(acc\_name,30);

break;

case '2':

cout<<"CURRENT NAME OF THE NOMINEE: " << nominee;

cout<<"\nENTER THE UPDATED NOMINEE: ";

cin.ignore();

cin.getline(nominee,30);

break;

case '3':

cout<<"CURRENT CONTACT NUMBER: " << contact\_no;

int contact\_ok = 0;

char tmp\_contact[10];

while (contact\_ok== 0)

{

cout<<"\nENTER CONTACT PHONE NUMBER (10 digits): ";

cin.clear();

cin >> tmp\_contact;

contact\_ok = 1;

for (int i=0; i < 10; i++)

{

if(!(isdigit(tmp\_contact[i] )))

{

cout << "PLEASE ENTER A VALID PHONE NUMBER - ONLY DIGITS 0 to 9!" << endl;

contact\_ok = 0;

break;

}

}

}

strcpy(contact\_no, tmp\_contact);

contact\_no[10] = '\0';

break;

}

}

//FUNCTION TO DISPLAY THE ACCOUNT DETAILS

void txn\_account::show\_acc\_details()

{

cout<<"\nNAME OF THE ACCOUNT HOLDER:"<<acc\_name<<endl;

cout<<"NAME OF THE NOMINEE:"<<nominee<<endl;

cout<<"ACCOUNT HOLDER CONTACT NUMBER:"<<contact\_no;

if (is\_active)

{

cout<<"\nACCOUNT STATUS: ACTIVE\n";

}

else

{

cout<<"\nACCOUNT STATUS: INACTIVE\n";

}

if (acc\_type == 'S')

{

cout<<"ACCOUNT TYPE: SAVING ACCOUNT";

}

else

{

cout<<"ACCOUNT TYPE: CURRENT ACCOUNT";

}

cout<<"\nACCOUNT BALANCE: "<<deposit<<endl;

}

//FUNCTION TO RETURN ACCOUNT NUMBER

int txn\_account::get\_acc\_number()

{

return acc\_no;

}

//FUNCTION TO RETURN ACCOUNT NAME

void txn\_account::get\_acc\_name(char \*name)

{

strcpy (name,acc\_name);

}

//FUNCTION TO RETURN ACCOUNT TYPE

char txn\_account::get\_type()

{

return acc\_type;

}

//FUNCTION TO RETURN ACCOUNT STATUS; 1 = Active; 0 = Inactive

int txn\_account::acc\_active()

{

return is\_active;

}

//FUNCTION TO RETURN ACCOUNT BALANCE

int txn\_account::get\_balance()

{

return deposit;

}

//FUNCTION TO ADD DEPOSIT AMOUNT

void txn\_account::deposit\_amount(int amount)

{

deposit += amount;

}

//FUNCTION TO WITHDRAW AMOUNT

int txn\_account::withdraw\_amount(int amount)

{

int reduced\_flag = 0;

//Check for account type and minimum balance to allow withdrawal

if (acc\_type == 'S')

{

if ( (deposit - amount) < 500 )

{

cout << "\nCANNOT WITHDRAW THIS AMOUNT AS THE MINIMUM BALANCE REQUIRED IS Rs. 500";

}

else

{

deposit -= amount;

reduced\_flag = 1;

}

}

else

{

if ( (deposit - amount) < 1000 )

{

cout << "\nCANNOT WITHDRAW THIS AMOUNT AS THE MINIMUM BALANCE REQUIRED IS Rs. 1000";

}

else

{

deposit -= amount;

reduced\_flag = 1;

}

}

return reduced\_flag;

}

//FUNCTION TO DISABLE THE ACCOUNT

void txn\_account::disable\_account()

{

is\_active = 0;

}

/\*----------------------------------------------------------------------

Definition of functions of the class: loan\_account

----------------------------------------------------------------------\*/

//FUNCTION TO POPULATE THE LOAN ACCOUNT DETAILS

void loan\_account::create\_account()

{

//Assign account number automatically

acc\_no = get\_next\_loanacc();

//The account is set to active by default

is\_active = 1;

cout<<"\nENTER THE LOAN ACCOUNT HOLDER NAME: ";

cin.ignore();

cin.getline(acc\_name,30);

cout<<"\nENTER THE GUARANTOR FOR THE ACCOUNT: ";

cin.clear();

cin.getline(guarantor,30);

//Get the contact number and validate that it is all digits

int contact\_ok = 0;

char tmp\_contact[10];

while (contact\_ok== 0)

{

cout<<"\nENTER CONTACT PHONE NUMBER (10 digits): ";

cin.clear();

cin >> tmp\_contact;

contact\_ok = 1;

for (int i=0; i < 10; i++)

{

if(!(isdigit(tmp\_contact[i] )))

{

cout << "PLEASE ENTER A VALID PHONE NUMBER - ONLY DIGITS 0 to 9!" << endl;

contact\_ok = 0;

break;

}

}

}

strcpy(contact\_no, tmp\_contact);

contact\_no[10] = '\0';

//Get the initial loan amount; minimum amount is Rs. 10,000

loan\_balance = 0;

while ( loan\_balance < 10000)

{

cout<<"ENTER THE SANCTIONED LOAN AMOUNT (Min Rs. 10,000): ";

cin >> loan\_balance;

}

}

//FUNCTION TO MODIFY LOAN ACCOUNT DETAILS

void loan\_account::modify\_account()

{

char option;

while(option !='1' && option !='2' && option !='3' )

{

clrscr();

cout<<"\nOPTIONS TO MODIFY THE ACCOUNT DETAILS:";

cout<<"\n1.CHANGE LOAN ACCOUNT HOLDER NAME";

cout<<"\n2.CHANGE LOAN GUARANTOR NAME";

cout<<"\n3.CHANGE CONTACT NUMBER";

cout<<"\nENTER OPTION:";

cin>>option;

}

switch(option)

{

case '1':

cout<<"CURRENT NAME OF THE LOAN ACCOUNT HOLDER: " << acc\_name;

cout<<"\nENTER THE UPDATED NAME: ";

cin.ignore();

cin.getline(acc\_name,30);

break;

case '2':

cout<<"CURRENT NAME OF THE GUARANTOR: " << guarantor;

cout<<"\nENTER THE UPDATED GUARANTOR: ";

cin.ignore();

cin.getline(guarantor,30);

break;

case '3':

cout<<"CURRENT CONTACT NUMBER: " << contact\_no;

int contact\_ok = 0;

char tmp\_contact[10];

while (contact\_ok== 0)

{

cout<<"\nENTER CONTACT PHONE NUMBER (10 digits): ";

cin.clear();

cin >> tmp\_contact;

contact\_ok = 1;

for (int i=0; i < 10; i++)

{

if(!(isdigit(tmp\_contact[i] )))

{

cout << "PLEASE ENTER A VALID PHONE NUMBER - ONLY DIGITS 0 to 9!" << endl;

contact\_ok = 0;

break;

}

}

}

strcpy(contact\_no, tmp\_contact);

contact\_no[10] = '\0';

break;

}

}

//FUNCTION TO DISPLAY THE LOAN ACCOUNT DETAILS

void loan\_account::show\_acc\_details()

{

cout<<"\nNAME OF THE LOAN ACCOUNT HOLDER:" << acc\_name<<endl;

cout<<"NAME OF THE GUARANTOR:" << guarantor << endl;

cout<<"LOAN ACCOUNT HOLDER CONTACT NUMBER:"<<contact\_no << endl;

if (is\_active)

{

cout<<"LOAN ACCOUNT STATUS: ACTIVE";

}

else

{

cout<<"LOAN ACCOUNT STATUS: CLOSED";

}

cout<<"\nLOAN BALANCE: "<<loan\_balance<<endl;

}

//FUNCTION TO RETURN LOAN ACCOUNT NUMBER

int loan\_account::get\_acc\_number()

{

return acc\_no;

}

//FUNCTION TO RETURN ACCOUNT NAME

void loan\_account::get\_acc\_name(char \*name)

{

strcpy (name,acc\_name);

}

//FUNCTION TO RETURN ACCOUNT STATUS

int loan\_account::acc\_active()

{

return is\_active;

}

//FUNCTION TO RETURN LOAN ACCOUNT BALANCE

int loan\_account::get\_balance()

{

return loan\_balance;

}

//FUNCTION TO REDUCE LOAN BALANCE

void loan\_account::reduce\_loan\_balance (int amount)

{

loan\_balance -= amount;

}

//FUNCTION TO ADD/INCREASE LOAN AMOUNT

void loan\_account::increase\_loan\_balance(int amount)

{

loan\_balance += amount;

}

//FUNCTION TO DISABLE THE ACCOUNT

void loan\_account::disable\_account()

{

is\_active = 0;

}

/\*----------------------------------------------------------------------

Declaration of the functions

----------------------------------------------------------------------\*/

void welcome (); // Function to display welcome screen

void create\_new\_account(); // Function to create a new account

void deposit\_amount(); // Function to add deposit amount

void withdraw\_amount(); // Function to withdraw amount

void repay\_loan(); //Function to repay loan amount

void add\_loan\_amount(); // Function to add loan amount

void modify\_account(); // Function to modify account details

void display\_account\_details(); //Function to call display account functions

void display\_txn\_account(int); //Function to search and display transaction account

void display\_loan\_account (int);//Function to search and display Loan account

void display\_all\_accounts(); //Function to display the details of all accounts

void close\_account(); //Function to disable/close an account

void write\_txn\_account(txn\_account); // Function to write a transaction account to file

void write\_loan\_account(loan\_account); // Function to write a loan account to file

/\*----------------------------------------------------------------------

Main Function

----------------------------------------------------------------------\*/

int main()

{

char option;

//Call the function to display the welcome screen

welcome();

//Display the options and capture the option selected by user

while(option!= 'X')

{

clrscr();

cout<<"\n\t\t\tBANK ACCOUNT MANAGEMENT AND TRANSACTION OPTIONS";

cout<<"\n1.CREATE AN ACCOUNT";

cout<<"\n2.SAVINGS/CURRENT ACCOUNT: DEPOSIT AMOUNT";

cout<<"\n3.SAVINGS/CURRENT ACCOUNT: WITHDRAW AMOUNT";

cout<<"\n4.LOAN ACCOUNT: REPAY LOAN";

cout<<"\n5.LOAN ACCOUNT: ADD ADDITIONAL LOAN AMOUNT";

cout<<"\n6.MODIFY ACCOUNT DETAILS";

cout<<"\n7.SHOW ACCOUNT DETAILS";

cout<<"\n8.DISPLAY ALL ACCOUNT HOLDERS LIST";

cout<<"\n9.CLOSE AN ACCOUNT";

cout<<"\nX.EXIT";

cout<<"\nENTER OPTION:";

cin>>option;

option = toupper(option);

switch(option)

{

case '1':

//Call the function to create a new account

create\_new\_account();

break;

case '2':

//Call the function to deposit amount in a savings/current account

deposit\_amount();

break;

case '3':

//Call the function to withdraw amount from a savings/current account

withdraw\_amount();

break;

case '4':

//Call the function to repay loan amount

repay\_loan();

break;

case '5':

//Call the function to add loan amount

add\_loan\_amount();

break;

case '6':

//Call the function to modify account details for transaction/loan account

modify\_account();

break;

case '7':

//Call the function to display the details of a specific account

display\_account\_details();

break;

case '8':

//Call the function to display the details of all accounts

display\_all\_accounts();

break;

case '9':

//Call the function to close/disable an account

close\_account();

break;

case 'X':

//Exit the program

cout<<"EXITING........";

break;

default:

break;

}

}

return 0;

}

/\*----------------------------------------------------------------------

Definition of the functions used in the program

----------------------------------------------------------------------\*/

// Function to return the next Transaction account number to be used for a new account

int get\_next\_txnacc()

{

ifstream t\_file;

txn\_account t\_account;

int tmp\_acc\_num = 0;

t\_file.open("txn\_account.dat",ios::binary);

if (t\_file.fail())

{

//There is no account file; hence return the initial account number

return 1000;

}

while (t\_file.read((char \*)&t\_account, sizeof(txn\_account)))

{

tmp\_acc\_num = t\_account.get\_acc\_number();

}

t\_file.close();

// Assign a default starting value for the account, if this is the first account

if (tmp\_acc\_num == 0)

{

tmp\_acc\_num = 1000;

}

else // Increment the account number and return

{

tmp\_acc\_num++;

}

return tmp\_acc\_num;

}

// Function to return the next Loan account number to be used for a new account

int get\_next\_loanacc()

{

ifstream l\_file;

loan\_account l\_account;

int tmp\_acc\_num = 0;

l\_file.open("loan\_account.dat",ios::binary);

if (l\_file.fail())

{

//There is no account file; hence return the initial account number

return 5000;

}

while (l\_file.read((char \*)&l\_account, sizeof(loan\_account)))

{

tmp\_acc\_num = l\_account.get\_acc\_number();

}

l\_file.close();

// Assign a default starting value for the account, if this is the first account

if (tmp\_acc\_num == 0)

{

tmp\_acc\_num = 5000;

}

else // Increment the account number and return

{

tmp\_acc\_num++;

}

return tmp\_acc\_num;

}

// Function to display the welcome message

void welcome ()

{

clrscr();

cout<<"\n\n\n";

cout<<"\t\t\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n";

cout<<"\n";

cout<<"\t\t\t\tFINANCIAL\n\t\t\tACCOUNT MANAGEMENT SYSTEM\n";

cout<<"\n";

cout<<"\t\t\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n";

cout<<"\n\n";

cout<<"\t\tDONE BY:";

cout<<"\n\t\tAISHWARYA AND SAHITHYA";

cout<<"\n\n\n\t\tPRESS ENTER TO CONTINUE...";

getch();

clrscr();

}

//Function to create a new account

void create\_new\_account()

{

char type;

//Get the account type - Transaction or Loan

while (type != 'T' && type != 'L')

{

cout<<"\nENTER THE ACCOUNT TYPE(T=TRANSACTION; L=LOAN):";

cin >> type;

type = toupper(type);

}

// Create a Transaction account

if ( type == 'T')

{

txn\_account t\_account;

t\_account.create\_account(); // Populate the class object with account details

write\_txn\_account(t\_account); // write the account object to the file

}

else

{

loan\_account l\_account;

l\_account.create\_account(); // Populate the class object with account details

write\_loan\_account(l\_account); // write the account object to the file

}

}

//Function to write transaction account details to file

void write\_txn\_account(txn\_account t\_account)

{

ofstream t\_file;

t\_file.open("txn\_account.dat",ios::binary|ios::app);

if (t\_file.fail())

{

cout <<"Error: The file txn\_account.dat could not be opened";

return;

}

t\_file.write((char \*)&t\_account, sizeof(t\_account));

t\_file.close();

cout <<"NEW TRANSACTION ACCOUNT: " << t\_account.get\_acc\_number() << " SUCCESSFULLY CREATED\n";

t\_file.close();

cout<<"\nPRESS ENTER TO CONTINUE...";

getch();

}

//Function to write loan account details to file

void write\_loan\_account(loan\_account l\_account)

{

ofstream l\_file;

l\_file.open("loan\_account.dat",ios::binary|ios::app);

if (l\_file.fail())

{

cout <<"Error: The file loan\_account.dat could not be opened";

return;

}

l\_file.write((char \*)&l\_account, sizeof(loan\_account));

l\_file.close();

cout <<"NEW LOAN ACCOUNT: " << l\_account.get\_acc\_number() << " SUCCESSFULLY CREATED\n";

l\_file.close();

cout<<"PRESS ENTER TO CONTINUE...";

getch();

}

//Function to get the requirements to display account details and call respective functions

void display\_account\_details()

{

char type;

//Get the account type - Transaction or Loan

while (type != 'T' && type != 'L')

{

cout<<"\nENTER THE TYPE OF ACCOUNT TO DISPLAY (T=TRANSACTION; L=LOAN):";

cin >> type;

type = toupper(type);

}

// Retrieve and Display the account details

if ( type == 'T')

{

int t\_acc\_num;

cout<<"\nENTER THE TRANSACTION ACCOUNT NUMBER TO SEARCH AND DISPLAY:";

cin >> t\_acc\_num;

display\_txn\_account(t\_acc\_num);

}

else

{

int l\_acc\_num;

cout<<"\nENTER THE LOAN ACCOUNT NUMBER TO SEARCH AND DISPLAY:";

cin >> l\_acc\_num;

display\_loan\_account(l\_acc\_num);

}

cout << "PRESS ENTER TO CONTINUE...";

getch();

}

//Function to search for transaction account from file and display

void display\_txn\_account (int acc\_num)

{

ifstream t\_file;

txn\_account t\_account;

int found = 0;

t\_file.open("txn\_account.dat",ios::binary);

if (t\_file.fail())

{

cout << "Error: The file txn\_account.dat could not be opened!";

return;

}

while ((t\_file.read((char \*)&t\_account, sizeof(txn\_account))) && (found == 0))

{

if (acc\_num == t\_account.get\_acc\_number())

{

found = 1;

cout << "\n--------------------------------------------------------------------------\n";

t\_account.show\_acc\_details();

cout << "\n--------------------------------------------------------------------------\n";

}

}

if (found == 0)

{

cout << " THE SPECIFIED ACCOUNT IS NOT FOUND!";

}

t\_file.close();

}

//Function to search for Loan account from file and display

void display\_loan\_account (int acc\_num)

{

ifstream l\_file;

loan\_account l\_account;

int found = 0;

l\_file.open("loan\_account.dat",ios::binary);

if (l\_file.fail())

{

cout << "Error: The file loan\_account.dat could not be opened!";

return;

}

while ((l\_file.read((char \*)&l\_account, sizeof(loan\_account))) && (found == 0))

{

if (acc\_num == l\_account.get\_acc\_number())

{

found = 1;

cout << "\n--------------------------------------------------------------------------\n";

l\_account.show\_acc\_details();

cout << "\n--------------------------------------------------------------------------\n";

}

}

if (found == 0)

{

cout << " THE SPECIFIED LOAN ACCOUNT IS NOT FOUND!";

}

l\_file.close();

}

void display\_all\_accounts()

{

char type;

//Get the account type option to display accounts

while (type != 'T' && type != 'L' && type != 'B')

{

cout<<"\nENTER THE TYPE OF ACCOUNTS TO DISPLAY (T=TRANSACTION; L=LOAN; B=BOTH):";

cin >> type;

type = toupper(type);

}

// Retrieve and Display the account details for Transaction accounts

if ( type == 'T' || type == 'B')

{

ifstream t\_file;

txn\_account t\_account;

char tmp\_acc\_name[30];

t\_file.open("txn\_account.dat",ios::binary);

if (t\_file.fail())

{

cout << "THERE ARE NO TRANSACTION ACCOUNTS FOUND!"<<endl;

}

else

{

// Display the header for the report

cout << "DETAILS OF TRANSACTION ACCOUNTS:\n";

cout<<"\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n";

cout<<setw(10)<<"ACC NO" << setw(30)<< " ACC HOLDER NAME" << setw(10) << " TYPE "<< setw(10) << " STATUS" << setw(10) << " ACC\_BALANCE\n" << setw(10);

cout<<"\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n";

// Read the records and display details

while (t\_file.read((char \*)&t\_account, sizeof(txn\_account)))

{

{

t\_account.get\_acc\_name(tmp\_acc\_name);

cout << setw(10)<<t\_account.get\_acc\_number();

cout << setw(30)<<tmp\_acc\_name;

if (t\_account.get\_type() == 'S')

{

cout << setw(10)<<"SAVING";

}

else

{

cout << setw(10)<<"CURRENT";

}

if (t\_account.acc\_active())

{

cout <<setw(10)<<"ACTIVE";

}

else

{

cout <<setw(10)<<"INACTIVE";

}

cout << setw(10)<<t\_account.get\_balance()<<"\n";

}

}

}

}

// Retrieve and Display the account details for Loan accounts

if ( type == 'L' || type == 'B')

{

ifstream l\_file;

loan\_account l\_account;

char tmp\_acc\_name[30];

l\_file.open("loan\_account.dat",ios::binary);

if (l\_file.fail())

{

cout << "\nTHERE ARE NO LOAN ACCOUNTS FOUND!"<<endl;

}

else

{

// Display the header for the report

cout << "\nDETAILS OF LOAN ACCOUNTS:\n";

cout<<"\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n";

cout<<setw(10)<<"ACC NO" << setw(30)<< " ACC HOLDER NAME" << setw(10) << " STATUS" << setw(10) << " LOAN\_BALANCE\n" << setw(10);

cout<<"\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n";

// Read the records and display details

while (l\_file.read((char \*)&l\_account, sizeof(loan\_account)))

{

{

l\_account.get\_acc\_name(tmp\_acc\_name);

cout << setw(10) << l\_account.get\_acc\_number() ;

cout << setw(30) << tmp\_acc\_name;

if (l\_account.acc\_active())

{

cout << setw(10) <<"OPEN" ;

}

else

{

cout << setw(10) <<"CLOSED" ;

}

cout << setw(10) << l\_account.get\_balance() <<"\n";

}

}

}

}

cout << "PRESS ENTER TO CONTINUE...";

getch();

}

//Function to disable/close an account

void close\_account()

{

char type;

int found = 0;

int tmp\_acc\_num;

txn\_account t\_account;

fstream t\_file;

loan\_account l\_account;

fstream l\_file;

//Get the account type to close - Transaction or Loan

while (type != 'T' && type != 'L')

{

cout<<"\nENTER THE TYPE OF ACCOUNT TO CLOSE (T=TRANSACTION; L=LOAN):";

cin >> type;

type = toupper(type);

}

// Get the account number to be closed

cout<<"\nENTER THE ACCOUNT NUMBER TO BE CLOSED:";

cin.clear();

cin >> tmp\_acc\_num;

// Close a a Transaction account

if ( type == 'T')

{

t\_file.open("txn\_account.dat", ios::in | ios::out | ios::binary);

if (t\_file.fail())

{

cout << "Error: The file txn\_account.dat could not be opened!";

cout << "\nPRESS ENTER TO CONTINUE...";

getch();

return;

}

while ((!t\_file.eof()) && (found == 0))

{

t\_file.read((char \*)&t\_account, sizeof(t\_account));

if (tmp\_acc\_num == t\_account.get\_acc\_number())

{

found = 1;

t\_account.show\_acc\_details();

t\_account.disable\_account();

t\_file.seekg(0,ios::cur);

t\_file.seekp(t\_file.tellg() - sizeof(t\_account));

t\_file.write((char\*)&t\_account,sizeof(t\_account));

cout << "\n ACCOUNT IS DISABLED";

cout << "\nPRESS ENTER TO CONTINUE...";

getch();

}

}

if (found == 0)

{

cout << "\nTHE SPECIFIED ACCOUNT IS NOT FOUND!";

cout << "\nPRESS ENTER TO CONTINUE...";

getch();

}

t\_file.close();

}

else

{

l\_file.open("loan\_account.dat", ios::in | ios::out | ios::binary);

if (l\_file.fail())

{

cout << "Error: The file txn\_account.dat could not be opened!";

cout << "\nPRESS ENTER TO CONTINUE...";

getch();

return;

}

while ((l\_file.read((char \*)&l\_account, sizeof(loan\_account))) && (found == 0))

{

if (tmp\_acc\_num == l\_account.get\_acc\_number())

{

found = 1;

l\_account.show\_acc\_details();

l\_account.disable\_account();

l\_file.seekg(0,ios::cur);

l\_file.seekp(l\_file.tellg() - sizeof(l\_account));

l\_file.write((char\*)&l\_account,sizeof(loan\_account));

cout << "\n THE LOAN ACCOUNT IS CLOSED";

cout << "\nPRESS ENTER TO CONTINUE...";

getch();

}

}

if (found == 0)

{

cout << "\nTHE SPECIFIED LOAN ACCOUNT IS NOT FOUND!";

cout << "\nPRESS ENTER TO CONTINUE...";

getch();

}

l\_file.close();

}

}

//Function to deposit amount into Savings/Current account

void deposit\_amount()

{

// Get the account number to deposit amount

int tmp\_acc\_num;

int dep\_amount = 0;

cout<<"\nENTER THE ACCOUNT NUMBER TO DEPOSIT AMOUNT:";

cin >> tmp\_acc\_num;

txn\_account t\_account;

fstream t\_file;

int found = 0;

t\_file.open("txn\_account.dat", ios::in | ios::out | ios::binary);

if (t\_file.fail())

{

cout << "Error: The file txn\_account.dat could not be opened!";

cout << "\nPRESS ENTER TO CONTINUE...";

getch();

return;

}

while ((t\_file.read((char \*)&t\_account, sizeof(txn\_account))) && (found == 0))

{

if (tmp\_acc\_num == t\_account.get\_acc\_number())

{

found = 1;

t\_account.show\_acc\_details();

if (!t\_account.acc\_active()) //Inactive account; can not transact

{

cout << "\nTHE ACCOUNT IS INACTIVE; NO TRANSACTIONS ALLOWED";

}

else

{

while (dep\_amount <= 0)

{

cout<<"\nENTER THE AMOUNT TO BE DEPOSITED (GREATER THAN 0):";

cin >> dep\_amount;

}

t\_account.deposit\_amount(dep\_amount);

t\_file.seekg(0,ios::cur);

t\_file.seekp(t\_file.tellg() - sizeof(t\_account));

t\_file.write((char\*)&t\_account,sizeof(txn\_account));

cout << "\nDEPOSIT AMOUNT ADDED TO THE ACCOUNT";

}

cout << "\nPRESS ENTER TO CONTINUE...";

getch();

}

}

if (found == 0)

{

cout << "\nTHE SPECIFIED ACCOUNT IS NOT FOUND!";

cout << "\nPRESS ENTER TO CONTINUE...";

getch();

}

t\_file.close();

}

//Function to withdraw amount from Savings/Current account

void withdraw\_amount()

{

// Get the account number to deposit amount

int tmp\_acc\_num;

int withdraw\_amount = 0;

int reduced\_flag = 0;

cout<<"\nENTER THE ACCOUNT NUMBER TO WITHDRAW AMOUNT:";

cin >> tmp\_acc\_num;

txn\_account t\_account;

fstream t\_file;

int found = 0;

t\_file.open("txn\_account.dat", ios::in | ios::out | ios::binary);

if (t\_file.fail())

{

cout << "Error: The file txn\_account.dat could not be opened!";

cout << "\nPRESS ENTER TO CONTINUE...";

getch();

return;

}

while ((t\_file.read((char \*)&t\_account, sizeof(txn\_account))) && (found == 0))

{

if (tmp\_acc\_num == t\_account.get\_acc\_number())

{

found = 1;

t\_account.show\_acc\_details();

if (!t\_account.acc\_active()) //Inactive account; can not transact

{

cout << "\nTHE ACCOUNT IS INACTIVE; NO TRANSACTIONS ALLOWED";

}

else

{

while (withdraw\_amount <= 0)

{

cout<<"\nENTER THE AMOUNT TO BE WITHDRAW (GREATER THAN 0):";

cin >> withdraw\_amount;

}

reduced\_flag = t\_account.withdraw\_amount(withdraw\_amount);

if (reduced\_flag == 1)

{

t\_file.seekg(0,ios::cur);

t\_file.seekp(t\_file.tellg() - sizeof(t\_account));

t\_file.write((char\*)&t\_account,sizeof(txn\_account));

cout << "\nAMOUNT WITHDRAWN FROM THE ACCOUNT";

}

else

{

cout << "\nAMOUNT NOT WITHDRAWN FROM THE ACCOUNT";

}

}

}

}

if (found == 0)

{

cout << "\nTHE SPECIFIED ACCOUNT IS NOT FOUND!";

}

t\_file.close();

cout << "\nPRESS ENTER TO CONTINUE...";

getch();

}

//Function to repay and reduce the loan balance amount

void repay\_loan()

{

// Get the account number to reduce loan balance

int tmp\_acc\_num;

int reduce\_amount = 0;

cout<<"\nENTER THE ACCOUNT NUMBER TO REDUCE LOAN BALANCE:";

cin >> tmp\_acc\_num;

loan\_account l\_account;

fstream l\_file;

int found = 0;

l\_file.open("loan\_account.dat", ios::in | ios::out | ios::binary);

if (l\_file.fail())

{

cout << "Error: The file loan\_account.dat could not be opened!";

cout << "\nPRESS ENTER TO CONTINUE...";

getch();

return;

}

while ((l\_file.read((char \*)&l\_account, sizeof(loan\_account))) && (found == 0))

{

if (tmp\_acc\_num == l\_account.get\_acc\_number())

{

found = 1;

l\_account.show\_acc\_details();

if (!l\_account.acc\_active()) //Inactive account; can not transact

{

cout << "\nTHE ACCOUNT IS CLOSED; NO TRANSACTIONS ALLOWED";

}

else

{

while (reduce\_amount <= 0)

{

cout<<"\nENTER THE AMOUNT TO BE REDUCED FROM LOAN BALANCE(GREATER THAN 0):";

cin >> reduce\_amount;

}

l\_account.reduce\_loan\_balance(reduce\_amount);

l\_file.seekg(0,ios::cur);

l\_file.seekp(l\_file.tellg() - sizeof(l\_account));

l\_file.write((char\*)&l\_account,sizeof(loan\_account));

cout << "\nLOAN AMOUNT REDUCED FROM THE ACCOUNT";

}

}

}

if (found == 0)

{

cout << "\nTHE SPECIFIED ACCOUNT IS NOT FOUND!";

}

l\_file.close();

cout << "\nPRESS ENTER TO CONTINUE...";

getch();

}

//Function to add loan amount to an existing loan account

void add\_loan\_amount()

{

// Get the account number to deposit amount

int tmp\_acc\_num;

int add\_amount = 0;

cout<<"\nENTER THE ACCOUNT NUMBER TO ADD LOAN AMOUNT:";

cin >> tmp\_acc\_num;

loan\_account l\_account;

fstream l\_file;

int found = 0;

l\_file.open("loan\_account.dat", ios::in | ios::out | ios::binary);

if (l\_file.fail())

{

cout << "Error: The file loan\_account.dat could not be opened!";

cout << "\nPRESS ENTER TO CONTINUE...";

getch();

return;

}

while ((l\_file.read((char \*)&l\_account, sizeof(loan\_account))) && (found == 0))

{

if (tmp\_acc\_num == l\_account.get\_acc\_number())

{

found = 1;

l\_account.show\_acc\_details();

if (!l\_account.acc\_active()) //Inactive account; can not transact

{

cout << "\nTHE ACCOUNT IS CLOSED; NO TRANSACTIONS ALLOWED";

}

else

{

while (add\_amount <= 0)

{

cout<<"\nENTER THE LOAN AMOUNT TO BE ADDED (GREATER THAN 0):";

cin >> add\_amount;

}

l\_account.increase\_loan\_balance(add\_amount);

l\_file.seekg(0,ios::cur);

l\_file.seekp(l\_file.tellg() - sizeof(l\_account));

l\_file.write((char\*)&l\_account,sizeof(loan\_account));

cout << "\nLOAN AMOUNT ADDED TO THE LOAN ACCOUNT";

}

}

}

if (found == 0)

{

cout << "\nTHE SPECIFIED ACCOUNT IS NOT FOUND!";

}

l\_file.close();

cout << "\nPRESS ENTER TO CONTINUE...";

getch();

}

//Function to Modify Transaction or Loan account details

void modify\_account()

{

char type = 'X';

//Get the account type to modify - Transaction or Loan

while (type != 'T' && type != 'L')

{

cout<<"\nENTER THE TYPE OF ACCOUNT TO MODIFY (T=TRANSACTION; L=LOAN):";

cin >> type;

type = toupper(type);

}

// Get the account number to be closed

int tmp\_acc\_num;

cout<<"\nENTER THE ACCOUNT NUMBER TO BE MODIFIED:";

cin >> tmp\_acc\_num;

// Update the Transaction account

if ( type == 'T')

{

txn\_account t\_account;

fstream t\_file;

int found = 0;

t\_file.open("txn\_account.dat", ios::in | ios::out | ios::binary);

if (t\_file.fail())

{

cout << "Error: The file txn\_account.dat could not be opened!";

cout << "\nPRESS ENTER TO CONTINUE...";

getch();

return;

}

while ((t\_file.read((char \*)&t\_account, sizeof(txn\_account))) && (found == 0))

{

if (tmp\_acc\_num == t\_account.get\_acc\_number())

{

found = 1;

if (!t\_account.acc\_active()) //Inactive account; can not change

{

cout << "\nTHE ACCOUNT IS INACTIVE; NO UPDATES ALLOWED";

}

else

{

t\_account.modify\_account();

t\_file.seekg(0,ios::cur);

t\_file.seekp(t\_file.tellg() - sizeof(t\_account));

t\_file.write((char\*)&t\_account,sizeof(txn\_account));

cout << "\nACCOUNT DETAILS MODIFICATION COMPLETED";

}

}

}

if (found == 0)

{

cout << "\nTHE SPECIFIED ACCOUNT IS NOT FOUND!";

}

t\_file.close();

cout << "\nPRESS ENTER TO CONTINUE...";

getch();

}

else

{

loan\_account l\_account;

fstream l\_file;

int found = 0;

l\_file.open("loan\_account.dat", ios::in | ios::out | ios::binary);

if (l\_file.fail())

{

cout << "Error: The file txn\_account.dat could not be opened!";

cout << "\nPRESS ENTER TO CONTINUE...";

getch();

return;

}

while ((l\_file.read((char \*)&l\_account, sizeof(loan\_account))) && (found == 0))

{

if (tmp\_acc\_num == l\_account.get\_acc\_number())

{

found = 1;

if (!l\_account.acc\_active()) //Inactive account; can not change

{

cout << "\nTHE ACCOUNT IS CLOSED; NO UPDATES ALLOWED";

}

else

{

l\_account.modify\_account();

l\_file.seekg(0,ios::cur);

l\_file.seekp(l\_file.tellg() - sizeof(l\_account));

l\_file.write((char\*)&l\_account,sizeof(loan\_account));

cout << "\n THE LOAN ACCOUNT DETAILS MODIFICATION COMPLETED";

}

}

}

if (found == 0)

{

cout << "\nTHE SPECIFIED LOAN ACCOUNT IS NOT FOUND!";

}

l\_file.close();

cout << "\nPRESS ENTER TO CONTINUE...";

getch();

}

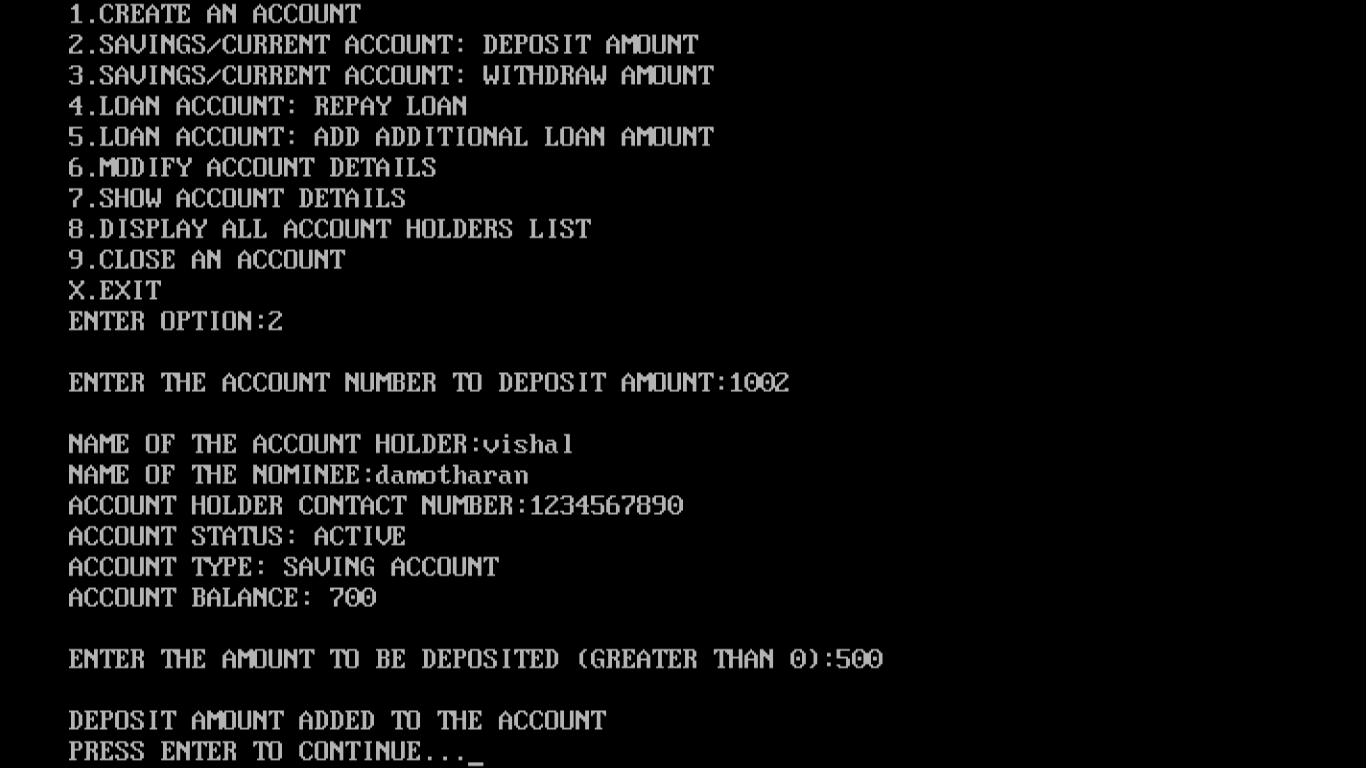
}

SAMPLE OUTPUTS:

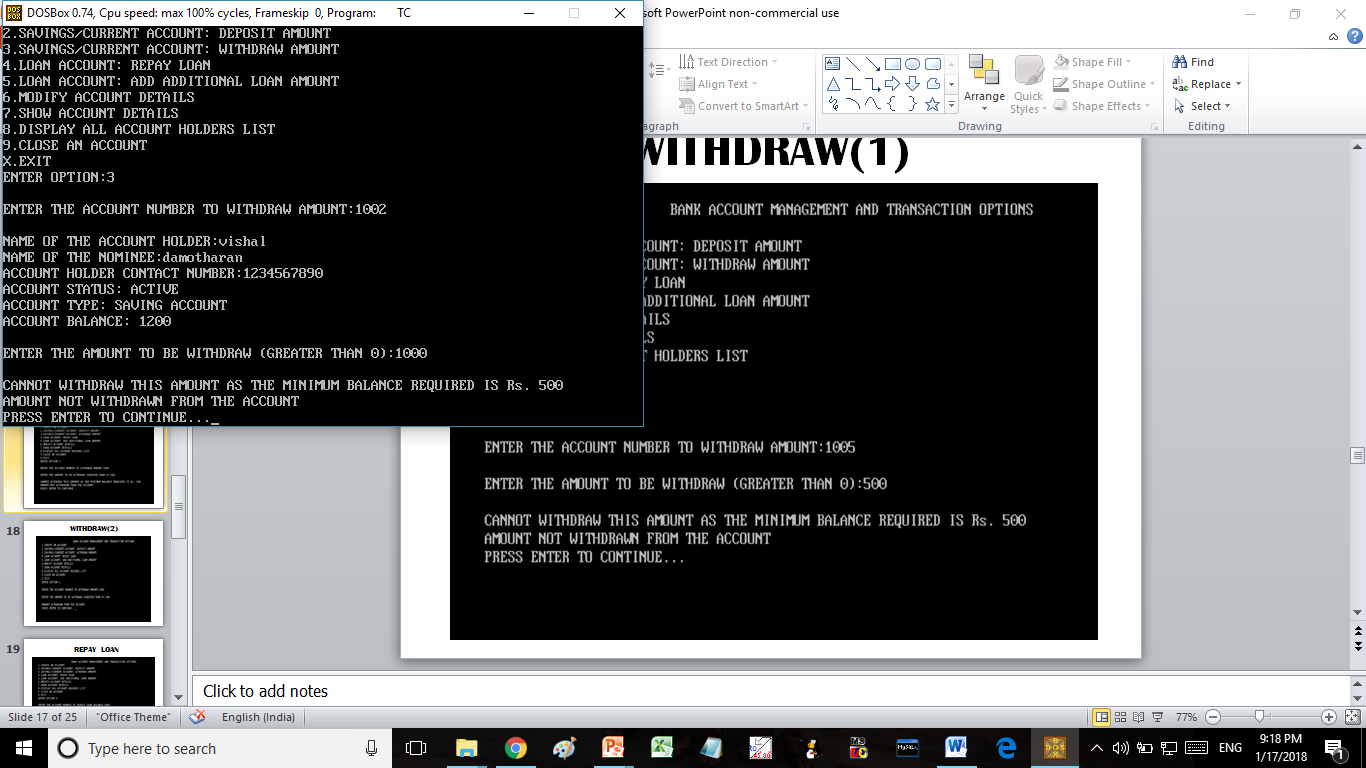
*CREATING AN ACCOUNT:*

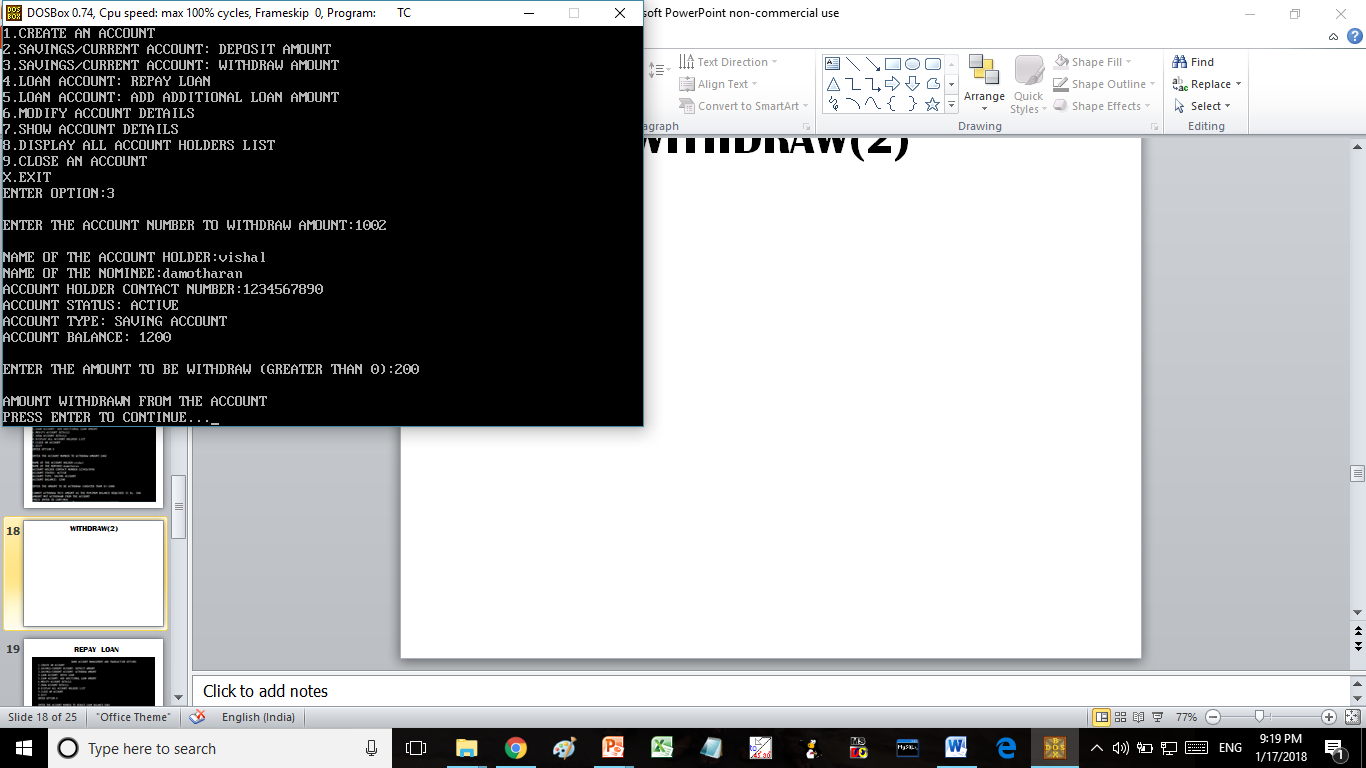


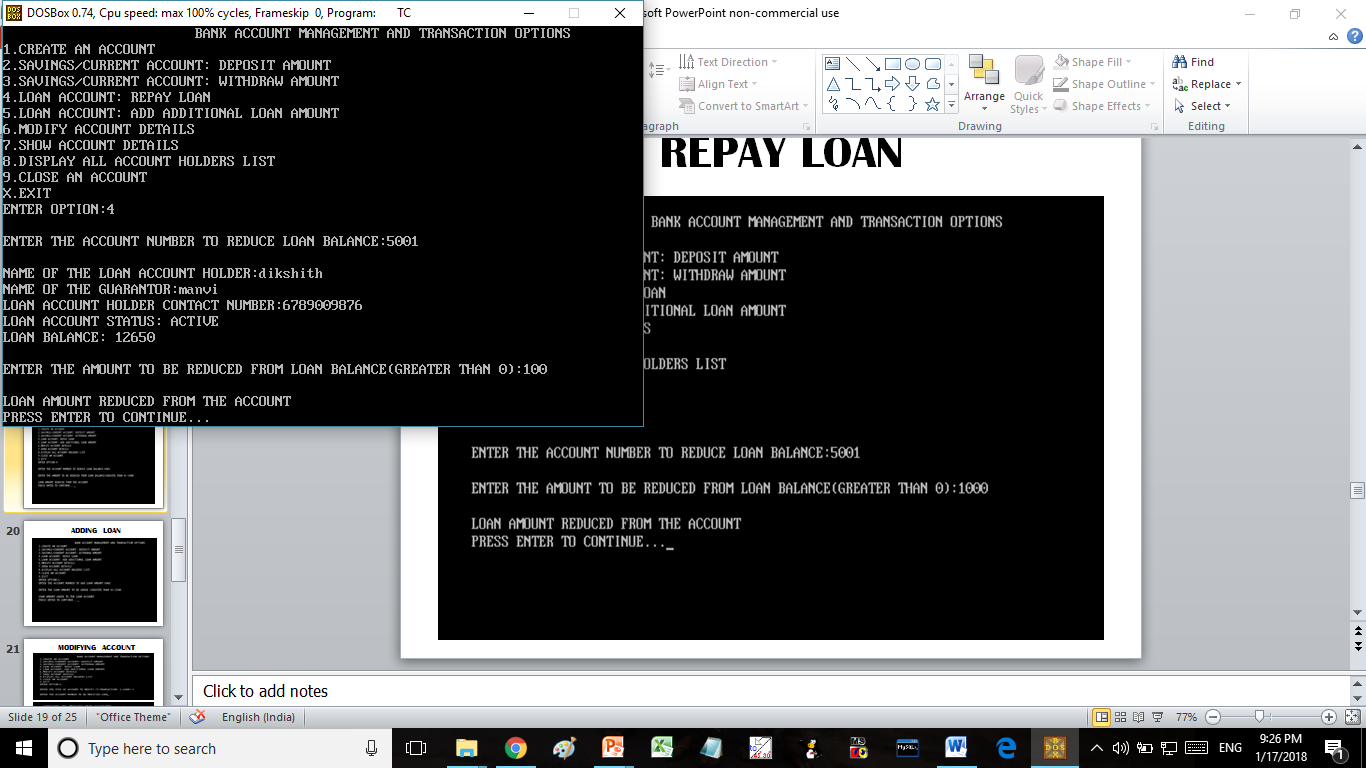
*DEPOSIT AMOUNT:*



*WITHDRAW AMOUNT:*



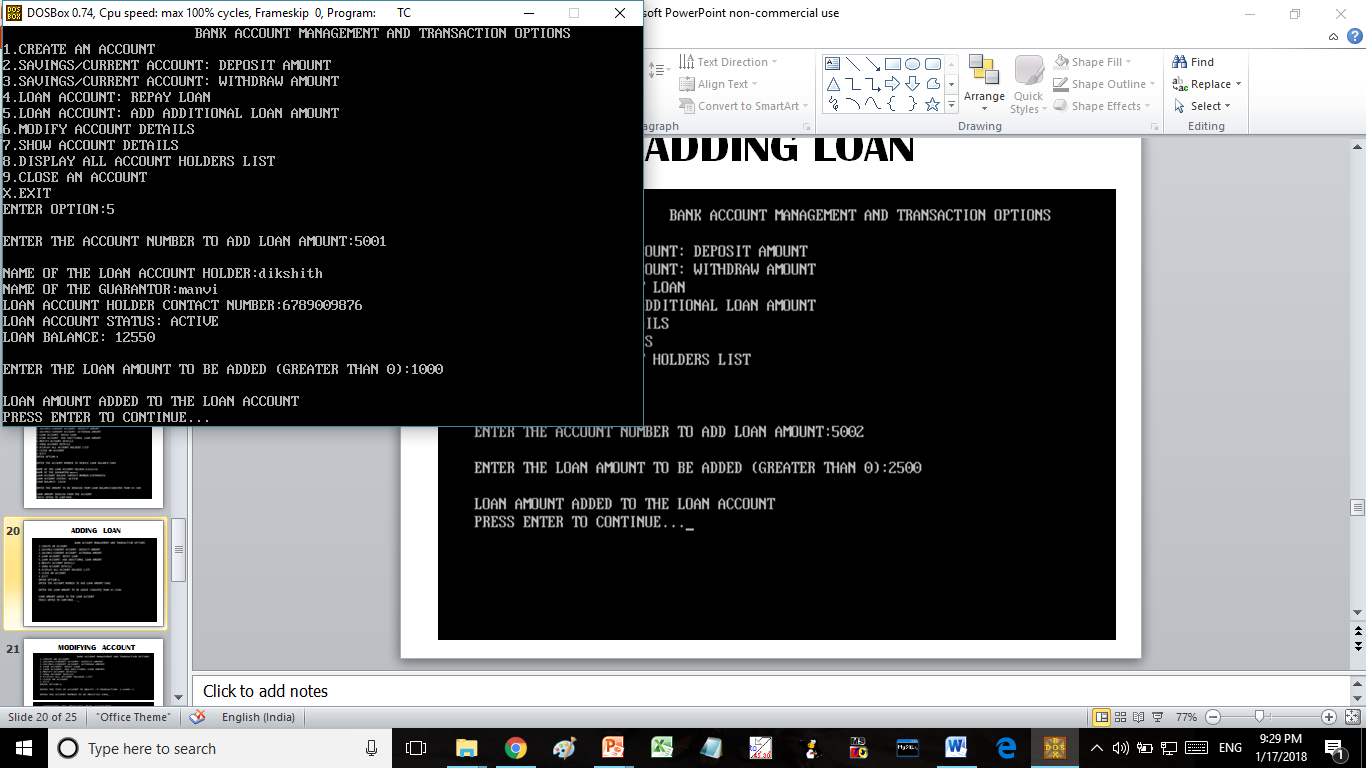




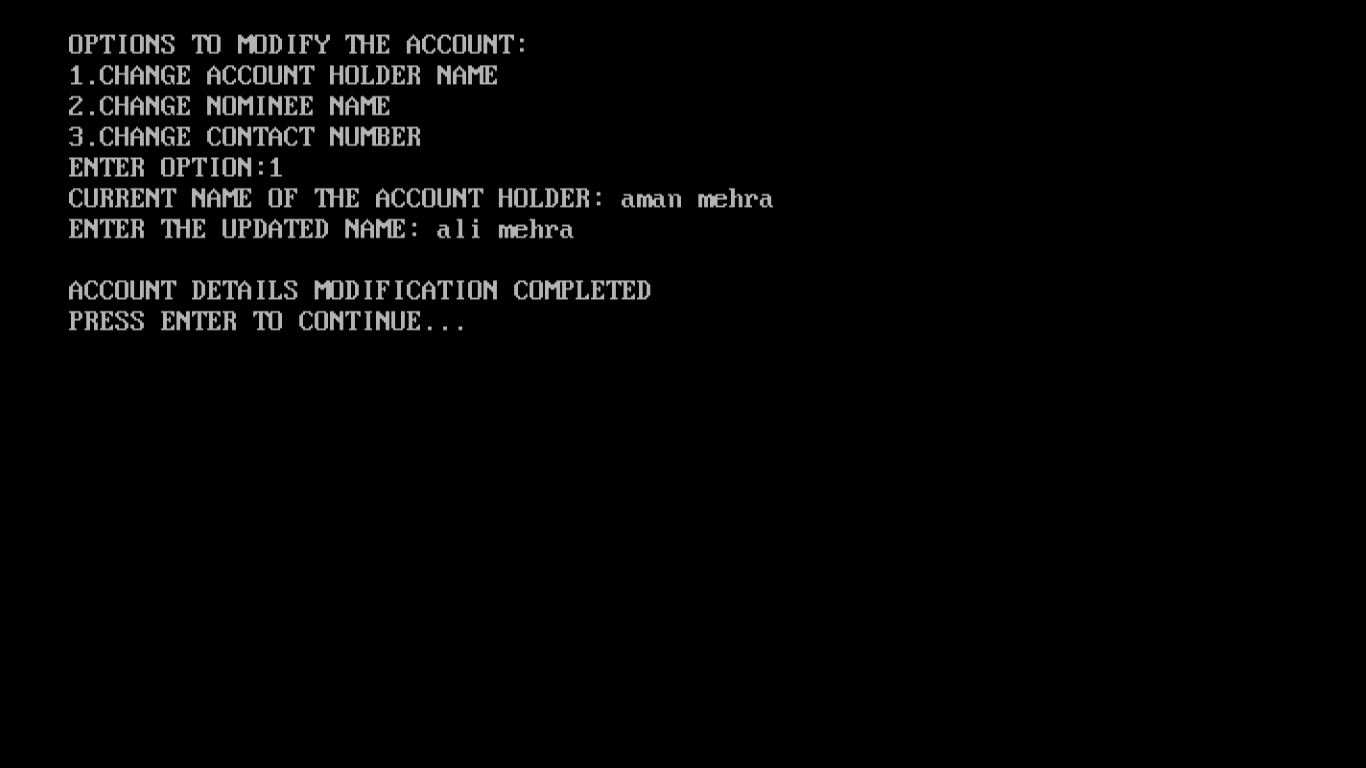
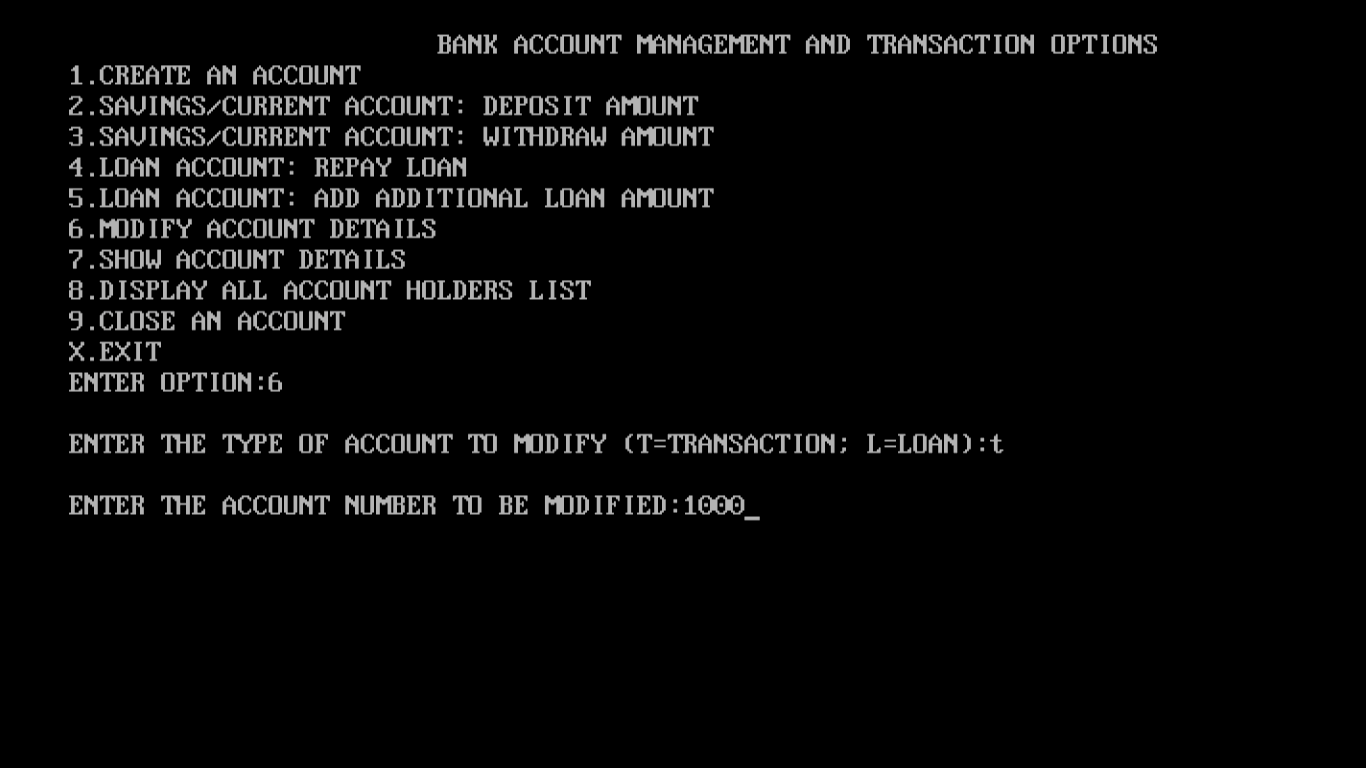
*REDUCE LOAN AMOUNT:*

*REDUCE LOAN AMOUNT:*

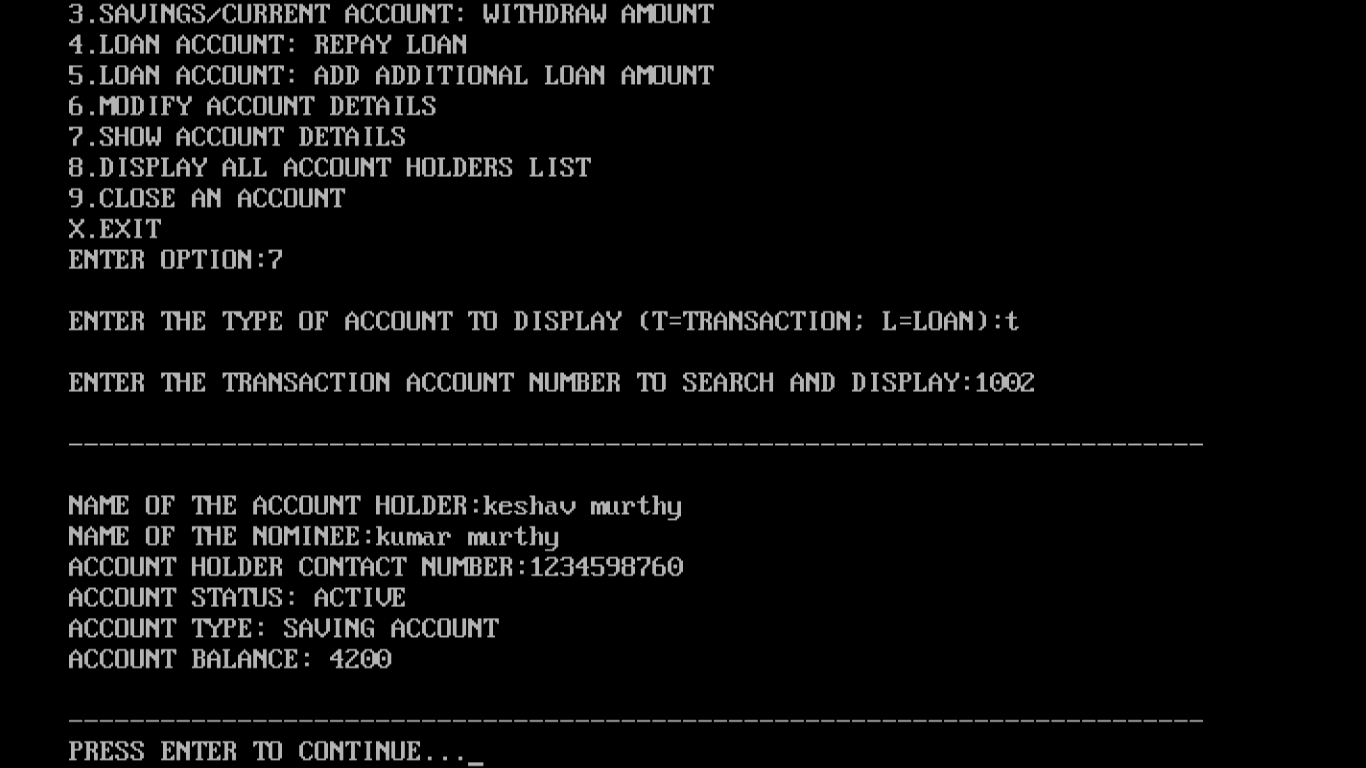
*ADD LOAN AMOUNT:*

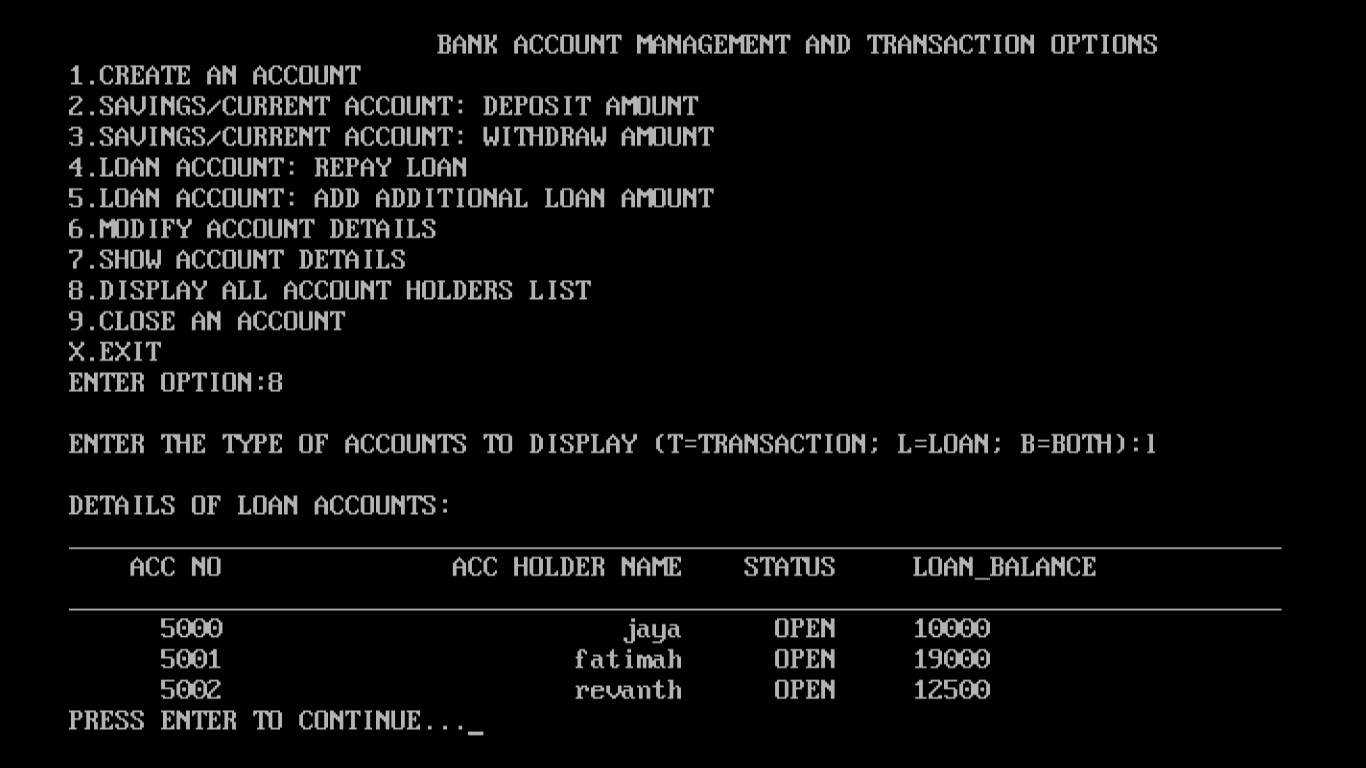


*MODIFY ACCOUNT:*



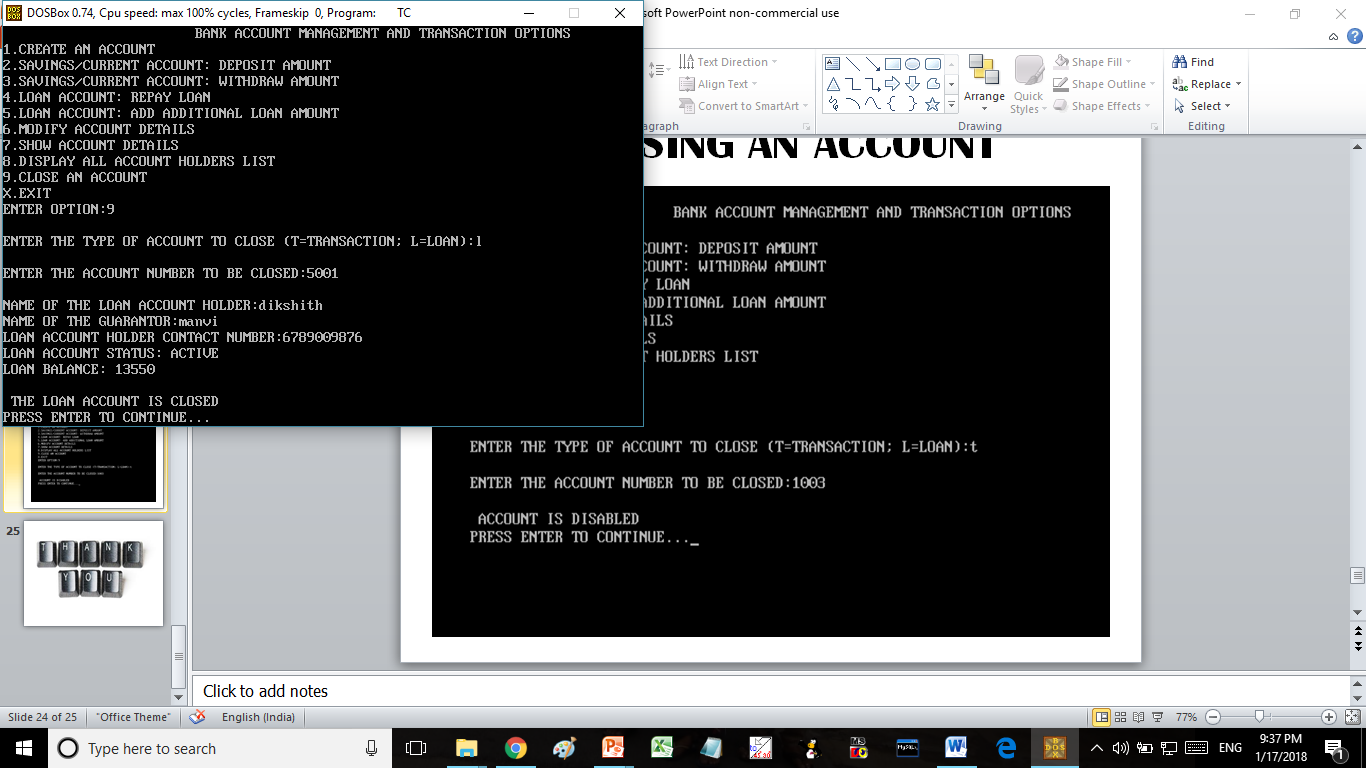
*SHOW ACCOUNT DETAILS:*





*ALL ACCOUNT HOLDER LIST:*

*CLOSE AN ACCOUNT:*



FUTURE ENHANCEMENTS:

The program could be enhanced to enhance the usability and ease of operations. Some of the enhancements that will be done in the future:

* A transactions log will be created for review and audit purposes
* A login module will be added for increased security
* A module will be added to allow the calculation of interest rates for different account types.

Conclusion:

The Financial Accounts Management System has been developed to make the job of maintaining financial accounts easier and efficient for the bank users. Using this application will help the bank users to create and maintain savings, current and loan accounts easily. They will also be able to complete the financial transactions faster. As the transactions are done using the program, manual transaction error will be eliminated.

References:

* Computer Science With C++ - by Sumita Arora
* C++ Reference web sites:
  + http://[www.programiz.com](http://www.programiz.com)
  + http://[www.cprogramming.com](http://www.cprogramming.com)
  + http://www.cplusplus.com
  + http://www.cppforschool.com